Claims

What is claimed is:

1. A graphics system comprising:

a memory configured to receive and store graphics data, wherein the memory comprises,

a RAM configured to store the graphics data,

a level two cache memory connected to the RAM, and

a level one cache memory connected to the level two cache memory;

an array of registers configured to store status information, wherein the status information tracks and indicates accesses to the graphics data in the level one cache, wherein the status information further indicates whether the graphics data is modified or unmodified; and

a memory request processor connected to the memory and to the array of registers, wherein the memory request processor controls the transfer of graphics data from the level one cache memory to the level two cache memory according to the status information.

2. The graphics system of claim 1, wherein the graphics data comprises samples.

3. The graphics system of claim 1, wherein the graphics data comprises pixels.

4. The graphics system of claim 1, wherein the level one cache memory is divided into logical blocks, and wherein each register of status information corresponds to one logical block.

5. The graphics system of claim 4, wherein the status information comprises:

a least recently used (LRU) count, wherein the LRU count indicates which logical block in the level one cache memory has been least recently accessed; and

a dirty block bit, wherein the dirty block bit indicates which portions of the graphics data in the level one cache memory has been modified.

6. The graphics system of claim 1, further comprising a request queue connected to the memory request processor, wherein the request queue comprises a first-in-first-out (FIFO) storage structure, wherein the request queue is configured to receive and buffer memory requests, and wherein the request queue is further configured to output the memory requests to the memory request processor in response to control signals from the memory request processor.

7. The graphics system of claim 6, wherein the array of registers is divided into two distinct sets, wherein one set of registers stores status information indicative of a current state of the level one cache, and wherein the second set of registers stores status information indicative of the current state of the level one cache plus the predicted results of one or more memory requests pending in the request queue.

8. The graphics system of claim 1, wherein the memory further comprises a shift register connected to the RAM, wherein the shift register is configured to receive and store portions of the graphics data from the RAM, and wherein the shift register is further configured to output graphics data serially in response to an external clock signal.

9. The graphics system of claim 8, further comprising a display device, wherein the display device displays images according to the graphics data.

10. The graphics system of claim 1, wherein the memory further comprises an arithmetic logic unit (ALU) connected to the level one cache memory, wherein the ALU is configured to:

receive as one operand graphics data from a source external to the memory;

receive as a second operand graphics data stored in the level one cache;

arithmetically combine the two operands according to a function defined by an external control signal; and

store the results of the arithmetic combination in the level one cache.

11. A graphics system comprising:

a memory configured to receive and store tiles of pixel data, wherein the memory comprises,

a RAM configured to store the tiles of pixel data,

a level two cache connected to the RAM, and

a level one cache connected to the level two cache memory;

a buffer configured to store a list of tags indicative of level one cache status; and

a cache controller connected to the memory and to the buffer, wherein the cache controller manages the transfer of graphics data from the level one cache to the level two cache according to the tags stored in the buffer.

12. The graphics system of claim 11, wherein each tag includes:

an LRU count, wherein the value of the LRU count indicates which tiles of pixel data stored in the level one cache have been least recently used; and

a dirty tag, wherein the contents of the dirty tag indicate which of the tiles of pixel data stored in the level one cache have been modified.

13. The graphics system of claim 11, wherein subsets of the pixel data stored in the level two cache are copied to the level one cache, and wherein the status information stored in the buffer indicates disparities between the pixel data stored in the level two cache and the associated copies of the pixel data stored in the level one cache.

Sub A1

10986747.020802

14. The graphics system of claim 11, wherein the cache controller further comprises a block cleanser, wherein the block cleanser periodically copies the pixel data stored in the level one cache to the level two cache according to the status information stored in the buffer.

15. The graphics system of claim 11, further comprising a bus interface, wherein the bus interface is configured to receive graphics data from a host system, and wherein the bus interface is further configured to reformat and communicate the graphics data to the graphics system, wherein the graphics system is configured to render the tiles of pixel data based on the graphics data.

16. The graphics system of claim 11, wherein the RAM comprises a plurality of DRAM banks, wherein each DRAM bank has one associated level two cache, wherein one level one cache is associated with a number of level two caches, and wherein the number of level two caches is greater than one.

17. A method for storing sample data in a memory array, wherein the method comprises:

a. arithmetically combining the sample data with the contents of a temporary storage space to form a result, wherein the result is stored in the temporary storage space;

b. maintaining a list of tag bits indicative of whether or not blocks of the sample data stored in the temporary storage space have been modified;

c. determining if any of the tag bits indicate that the corresponding blocks of sample data within the temporary storage space have been modified;

d. issuing a request to copy a block of sample data to the memory array from the temporary storage space in response to determining that the corresponding tag bits indicate that the sample data has been modified;

e. copying the modified block of sample data to the memory array from the temporary storage space; and

f. changing the state of the tag bits corresponding to the block of sample data copied to the memory array to indicate that the associated sample data is unmodified.

18. The method of claim 17, wherein (e) is allowed to execute only during empty memory cycles.

19. The method of claim 17, wherein (f) is executed each time sample data is transferred between the temporary storage space and the memory array independent of the direction of the transfer.

20. The method of claim 17, wherein (e) is accomplished in a single memory operation by performing a parallel transfer of sample data.

21. The method of claim 17, wherein (c), (d), (e), and (f) are forced to be executed in response to an urgent request for allocation of memory within the temporary storage space.

22. The method of claim 17, further comprising:

determining whether the sample data within the block of temporary storage space being examined is currently being accessed; and

delaying the execution of (d), (e) and (f) in response to detecting that the block of sample data within the temporary storage space is being accessed.

23. A method for externally managing cached pixel data within a 3D-RAM memory device, the method comprising:

a) identifying a block of memory within a level one cache in the 3D-RAM device, wherein the block is either a target or a source for a data block transfer with a level two cache in the 3D-RAM memory device;

b) maintaining a status word for each data block in the level one cache;

c) setting a dirty tag bit in one or more of the status words to a first state in response to a data block transfer, wherein the first state indicates that the data within the associated block of the level one cache is unmodified; and

d) identifying write operations to the level one cache, wherein the source of the write data is external to the 3D-RAM;

e) setting the dirty tag bit associated with the data value in the level one cache to a second state in response to a write operation, wherein the second state indicates that the data within the associated block of the level one cache is modified; and

f) synchronizing the data in the level two cache to the data in the level one cache according to the state information stored in the dirty tag bits in the status words.

24. The method of claim 23, wherein (f) is accomplished by transferring a block of data from the level one cache to the level two cache, wherein the block transferred is prioritized according to a least recently used (LRU) count in the status word, and wherein the LRU count is indicative of an activity level associated with the block.

25. The method of claim 23, wherein (e) is executed only if the data written to the level one cache in (d) is not equivalent to the data residing at the target location in the level one cache.

26. The method of claim 23, wherein (f) further comprises a write-through of the level two cache, wherein the write-through causes the data to also be written to a random access memory (RAM).

27. The method of claim 23, wherein the association of status words to blocks of level one cache memory is constant and not re-assignable.

28. The method of claim 23, wherein (f) is executed on a periodic basis.

29. The method of claim 23, further comprising:

setting a dirty tag bit in the status word in response to synchronizing the data in the level two cache with the associated block of data in the level one cache, wherein the dirty tag bit set indicates that the associated block of the level one cache memory is available for allocation.

30. A method for externally maintaining unlocked blocks of level one cache memory in a 3D-RAM device, wherein the unlocked blocks are free for allocation, the method comprising:

a) maintaining a list of access indicators, wherein each access indicator is uniquely associated with one block of level one cache memory;

b) maintaining a list of tag bits, wherein each tag bit is uniquely associated with one block of level one cache memory;

c) modifying the access indicators as a part of performing external data reads from and writes to the level one cache memory;

d) modifying the tag bits as a part of performing an external data write to the level one cache memory;

e) modifying the tag bits as part of issuing commands to transfer blocks of data between the level one cache and a level two cache;

f) periodically issuing commands to copy blocks of data from the level one cache to the level two cache according to the information stored in the associated tag bits and access indicators, and modifying the contents of the associated tag bits, wherein the contents of the tag bits are forced to indicate the associated block is unlocked.

31. The method of claim 30, wherein the information stored in the list of access indicators facilitates ordering the level one cache memory blocks chronologically according to external accesses to the level one cache memory blocks.

32. The method of claim 31, wherein level one cache memory blocks which are identified by the associated access indicators as being least recently used are given priority as part of (f).

H0085474 020280